Available online at www.sciencedirect.com
ScienceDirect
Information and Computation 00 (2015) 1–16
Procedia Computer Science

Position Discovery for a System of Bouncing Robots☆

Jurek Czyzowicz

Université du Québec en Outaouais.
Gatineau, Québec J8X 3X7, Canada.

Leszek Gąsieniec

University of Liverpool.
Liverpool L69 3BX, UK.

Adrian Kosowski

INRIA Bordeaux Sud-Ouest.
LaBRI, 33400 Talence, France

Evangelos Kranakis*

Carleton University.
Ottawa, Ontario K1S 5B6, Canada.
(613) 520-4333

Oscar Morales-Ponce

Chalmers University of Technology
Maskingränd 2, 412 58 Göteborg, Sweden

Eduardo Pacheco

Carleton University.
Ottawa, Ontario K1S 5B6, Canada.

Abstract

A collection of n anonymous mobile robots is deployed on a unit-perimeter ring or a unit-length line segment. Every robot starts moving at constant speed, and bounces each time it meets any other robot or segment endpoint, changing its walk direction. We study the problem of *position discovery*, in which the task of each robot is to detect the presence and the initial positions of all other robots. The robots cannot communicate or perceive information about the environment in any way other than by bouncing. Each robot has a clock allowing it to observe the times

☆This is an extended and revised version of a paper that appeared in the proceedings of the 26th International Symposium on Distributed Computing, DISC 2012, Salvador, Brazil, October 16-18, 2012. 341-355, Lecture Notes in Computer Science.

*Corresponding author

Email addresses: jurek.czyzowicz@uqo.ca (Jurek Czyzowicz), L.A.Gasienec@liverpool.ac.uk (Leszek Gąsieniec), kosowski@labri.fr (Adrian Kosowski), kranakis@scs.carleton.ca (Evangelos Kranakis), mooscar@chalmers.se (Oscar Morales-Ponce), epacheco@cmail.carleton.ca (Eduardo Pacheco)

of its bounces. The robots have no control over their walks, which are determined by their initial positions and the starting directions. Each robot executes the same *position detection algorithm*, which receives input data in real-time about the times of the bounces, and terminates when the robot is assured about the existence and the positions of all the robots.

Some initial configuration of robots are shown to be *infeasible* — no position detection algorithm exists for them. We give complete characterizations of all infeasible initial configurations for both the ring and the segment, and we design optimal position detection algorithms for all feasible configurations. For the case of the ring, we show that all robot configurations in which not all the robots have the same initial direction are feasible. We give a position detection algorithm working for all feasible configurations. The cost of our algorithm depends on the number of robots starting their movement in each direction. If the less frequently used initial direction is given to $k \leq n/2$ robots, the time until completion of the algorithm by the last robot is $\frac{1}{2}\lceil \frac{n}{k} \rceil$. We prove that this time is optimal. By contrast to the case of the ring, for the unit segment we show that the family of infeasible configurations is exactly the set of so-called *symmetric configurations*. We give a position detection algorithm which works for all feasible configurations on the segment in time 2, and this algorithm is also proven to be optimal.

Keywords: Position Detection Mobile Robots Distributed Computing Algorithms Complexity

1. Introduction

A mobile robot is an autonomous entity with the capabilities of *sensing*, i.e. ability to perceive some parameters of the environment, *communication* - ability to receive/transmit information to other robots, *mobility* - ability to move within the environment, and *computation* - ability to process the obtained data. Mobile robots usually act in a distributed way, i.e. a collection of mobile robots is deployed across the territory and they collaborate in order to achieve a common goal by moving, collecting and exchanging the data of the environment. The typical applications are mobile software agents (e.g. moving around and updating information about a dynamically changing network) or physical mobile robots (devices, robots or nano-robots, humans).

In many distributed applications, mobile robots operate in large collections of massively produced, cheap, tiny, primitive entities with very restricted communication, sensing and computational capabilities, mainly due to the limited production cost, size and battery power. Such groups of mobile robots, called *swarms*, often perform exploration or monitoring tasks in hazardous or hard to access environments. The usual swarm robot attributes assumed for distributed models include anonymity, negligible dimensions, no explicit communication, no common coordinate system (cf. [1]). Moreover, some of these models may assume obliviousness, limited visibility of the surrounding environment and asynchronous operation. In most situations involving such weak robots the fundamental research question concerns the feasibility of solving the given task (cf. [2, 3, 1]). When the question of efficiency is addressed, the cost of the algorithm is most often measured in terms of length of the robot's walk or the time needed to complete the task. This is also the case of the present paper, despite the fact that the robot does not have any control over its walk. In our case, the goal is to stop the robot's walk, imposed by the adversary, at the earliest opportunity - when the collected information (or its absence) is sufficient to produce the required solution.

There have been other papers studying theoretical aspects involving extremely limited robots in the context of their positioning in simple, uni-dimensional environments (e.g. [4, 5, 6]. One of the central problems in robotics concerns the related problem of multi-robot localization or positioning (cf. [7, 8, 9, 10, 11, 12, 13], where the robots use sensing devices (e.g. [11, 12]) or they leave and observe landmarks (e.g. [7, 9]) in order to exchange knowledge with other robots. The position discovery problem may be viewed as a version of mapping (cf. [14]), where the target is to discover and map of the static environment, rather than recognize the other robots.

There exists extensive literature in the robotics community on the related aspects of the problem, e.g. on robot-coordination, domain coverage and clean-up, target tracking, etc. [15] provide heuristics and a protocol to enable agents to negotiate and form coalitions (e.g., agents required to join together) when there is uncertain and heterogeneous information. [16] proves complexity results concerning the efficient use of

"ant robots" for covering a connected region on the Z^2 grid, whose area n is unknown in advance, but which is expanding at a given rate. For example, the minimum number of such robots is shown to be in $\Omega(\sqrt{n})$, while $\Theta(\sqrt{n})$ is sufficient when the region is expanding at a sufficiently slow rate. [17] studies the cooperative cleaners problem which requires that several agents clean a connected region of "dirty" pixels in Z^2. [18] studies team coordination for the collision free target tracking problem of multi-agent robot system. [19] present an approximation algorithm utilizing finite-horizon planning and implicit coordination (to achieve linear scalability in the number of searchers) for the problem of locating a mobile, non-adversarial target in an indoor environment using multiple robotic searchers. [20] is a feasibility study on building terrain-covering ant robots which leave trails in the terrain so as to cover a closed terrain repeatedly. [21] studies both theoretically and by simulation the behaviour of ant robots for one-time as well as repeated coverage of a terrain (e.g., for lawn mowing, mine sweeping, patrolling, etc).

Although the most frequently studied question for mobile robots is environment exploration, numerous papers related to such weak robots often study more basic tasks, such as pattern formation ([3, 22, 1, 23]). Gathering or point convergence ([24, 25]) and spreading (e.g. see [26]) also fall into this category. [1] introduced anonymous, oblivious, asynchronous, mobile robots which act in a so-called *look-compute-move* cycle. An important robot sensing capacity associated with this model permits to perceive the entire ([3, 22, 1]) or partial ([27, 25]) environment.

Contrary to the above model, in our paper, a robot has absolutely no control over its movement, which is determined by the bumps against its neighbors or the boundary points of the environment. In [28, 29] the authors introduced *population protocols*, modeling wireless sensor networks by extremely limited finite-state computational devices. The agents of population protocols also move according to some mobility pattern totally out of their control and they interact randomly in pairs. This is called *passive mobility*, intended to model, e.g., some unstable environment, like a flow of water, chemical solution, human blood, wind or unpredictable mobility of agents' carriers (e.g. vehicles or flocks of birds). In the recent work [30], a coordination mechanism based on meetings with neighboring robots on the ring was considered, also aiming at location discovery. The approach of [30] is randomized and the robots operate in the discrete environment in synchronous rounds.

Pattern formation is sometimes considered as one of the steps of more complex distributed task. Our involvement in the problem of this paper was motivated by the patrolling problem [31], where spreading the robots evenly around the environment may result in minimizing the *idleness* of patrolling, i.e., the time interval during which environment points remain unvisited by any robot. Clearly, position discovery discussed in the present paper is helpful in uniform spreading of the collection. A related problem was studied in [26], where the convergence rate of uniform spreading in a one-dimensional environment in synchronous and semi-synchronous settings was discussed. Previously, [32] studied the problem of n robots $\{0, 1 \ldots, n-1\}$, initially placed in arbitrary order on the ring. It was shown that the rule of each robot i moving to the middle point between $i-1$ and $i+1$ may fail to converge to equal spreading (it was also shown in [32] that the system would converge if a fair scheduler activates units sequentially).

The model adopted in our paper assumes robot anonymity, passive mobility (similarly to that adopted in [28, 29]), restricted local sensing through bounce perception with a neighbor robot only, no communication between the robots, and continuous time. The only ability of the robot is the tacit observation of the timing of bounces and the computation and reporting of robots' locations. The clock of each robot turns out to be a very powerful resource permitting to solve the problem efficiently in most cases.

2. The Model and Our Results

We consider a continuous, connected, one-dimensional universe in which the robots operate, which is represented either by a unit-perimeter ring or by a unit-length line segment. The ring is modeled by a real interval $[0, 1]$ with 0 and 1 corresponding to the same point. A set of n robots $r_0, r_1, \ldots, r_{n-1}$ is deployed in the environment and each of them starts moving at time $t = 0$ (where the indexing of the robots is used for purposes of analysis, only). The robots are not aware of the original positions and directions of other robots or the total number of robots in the collection. The robots move at constant unit speed, each robot starting its movement in one of the two directions. Each robot knows the perimeter of the ring (or the length of

the segment) and it has a clock permitting to register the time of each of its bounces and store it in its memory. All clocks are synchronized. We assume that the time and distance traveled are commensurable, so during time t each robot travels distance t. Consequently, in the paper we compare distances travelled to time intervals.

By $r_i(t) \in [0,1]$ we denote the position of robot r_i at time t. We suppose that originally each robot r_i occupies point $r_i(0)$ of the environment and that $0 \leq r_0(0) < r_1(0) < \ldots < r_{n-1}(0) < 1$. Each robot is given an initial direction (clockwise or counterclockwise in the ring and left-to-right or right-to-left on the segment) in which it starts its movement. By dir_i we denote the starting direction of robot r_i and we set $dir_i = 1$ if r_i starts its movement in the counterclockwise direction around the ring or the left-to-right direction along the segment. By $dir_i = -1$ we denote the clockwise starting direction (on the ring) or right-to-left (on the segment). We call the sequence of pairs $(r_0(0), dir_0), \ldots, (r_{n-1}(0), dir_{n-1})$ the *initial configuration* of robots.

When two robots meet, they *bounce*, i.e., they reverse the directions of their movements. We call the trajectory of a robot a *bouncing walk*. The robots have no control over their bouncing walks, which depend only on their initial positions and directions, imposed to them by an adversary, and the bounces caused by meeting other robots. Each robot has to report the coordinates of all robots of the collection, i.e., their initial positions and their initial directions. The robots cannot communicate in any other way except for observing their meeting times. Each robot is aware of the type of the environment (ring or segment). All robots are anonymous, i.e. they have to execute the same algorithm. The only information available to each robot is the *bounce sequence*, i.e. the series of time moments $t_1, t_2, \ldots$, corresponding to its bounces resulting from the meetings with other robots.

By *position detection algorithm* we mean a procedure executed by each robot, during which the robot performs its bouncing walk and uses its bounce sequence as the data of the procedure, outputting the initial positions and directions of all robots. By the *cost* $C_{\mathcal{A}}(n)$ of algorithm $\mathcal{A}$ we understand the smallest value, such that for any feasible initial configuration of n robots in the environment, each robot executing $\mathcal{A}$ can report the initial configuration while performing a bouncing walk of total distance $C_{\mathcal{A}}(n)$. As in some cases the cost of the algorithm varies, depending on the robots' initial directions, we denote by $C_{\mathcal{A}}(n,k)$ the cost of $\mathcal{A}$ for the class of initial configurations such that $1 \leq k \leq n/2$ robots start in one direction and $n-k$ start in the opposite one.

Question: Is it possible for each robot to find out, after some time of its movement, what is the number of robots in the collection and their relative positions in the environment? If not, what are the configurations of robots' initial positions and directions for which a position detection algorithm exists (i.e. it is possible to report the initial configuration after a finite time)? What is the smallest amount of time after which a robot is assured to identify all other robots in the collection?

Our goal is to propose an algorithm to be executed by any robot, which computes the original positions of all other robots of the collection. We say that such an algorithm is optimal if the time interval after which the robot is assured to have the knowledge of the positions of all other robots is the smallest possible.

We characterize all the feasible configurations for the ring and the segment. For both cases we give optimal position detection algorithms for all feasible configurations. Our algorithm for the segment requires $O(n)$ robot's memory, while constant size memory is sufficient for robots bouncing on the ring. [We suppose that in one memory word we may store a real value representing the robot's position in segment $[0,1]$.]

For the case of the ring, we show that all robot configurations with not all robots given the same initial direction are feasible. We give a position detection algorithm working for all feasible configurations. The cost of our algorithm is not constant, but it depends on the number of robots starting their movement in each direction. When $k \leq n/2$ is the number of robots starting their walks in one direction with $n-k$ given the opposite direction we prove that our algorithm has cost $\frac{1}{2}\lceil \frac{n}{k} \rceil$. We prove that this algorithm is optimal.

For the case of the segment we prove that no position detection algorithm exists for *symmetric* initial configurations. Each symmetric configuration is a configuration of a subset of robots on a subsegment, concatenated alternately with its reflected copy and itself. We give a position detection algorithm of cost 2 working for all feasible (non-symmetric) configurations on the segment. This algorithm is proven to be optimal.

In Section 3 we give the position detection algorithm for the ring and prove its correctness for all feasible

configurations. Section 4 analyses the cost of the position detection algorithm for the ring and proves its optimality. The segment environment is addressed in Section 5. The argument for the segment proceeds by reduction to that for the ring, but the criteria for a feasible configuration on the segment take a different form, dependent on the symmetry of the configuration.

3. The Algorithm on the Ring

As there is no system of coordinates on the ring common to all robots, each robot must compute the relative positions of other robots with respect to its own starting position. We may then infer that each robot assumes that its starting position is the point 0. We then suppose that $0 = r_0(0) < r_1(0) < \ldots < r_{n-1}(0) < 1$ and it is sufficient to produce the algorithm for robot r_0.

We assume in this paper that all robot indices are taken modulo n. When two robots meet, they reverse the directions of their movements, so the circular order of the robots around the ring never changes. When two robots r_i and r_{i+1} meet at time t, we have $r_i(t) = r_{i+1}(t)$, and $r_i(t)$ was moving counterclockwise while $r_{i+1}(t)$ was moving clockwise just before the meeting time t.

We denote by $dist(x, y)$ the distance that x has to traverse in the counterclockwise direction around the ring to reach the position of y (we call it the *counterclockwise distance* from x to y. Note that the *clockwise distance* from x to y equals $1 - dist(x, y)$.

In order to analyze the ring movement of the robots we consider an infinite line $L = (-\infty, \infty)$ and for each robot r_i, $0 \leq i \leq n-1$ we create an infinite number of its copies $r_i^{(j)}$, all having the same initial direction, such that their initial positions are $r_i^{(j)}(0) = j + r_i(0)$ for all integer values of $j \in \mathbb{Z}$ (see Fig. 1). We show that, when all copies of robots move along the infinite line while bouncing at the moments of meeting, all copies $r_i^{(j)}$ of a robot r_i bounce and reverse their movements at the same time. More precisely we prove

Lemma 1. *For all $t \geq 0, 0 \leq i \leq n-1$ and $j \in \mathbb{Z}$ we have $r_i^{(j+1)}(t) = r_i^{(j)}(t) + 1$.*

Proof. Since the claim of the lemma holds by construction at time $t = 0$ and at any bounce moment all copies of the bouncing robots $r_b^{(j)}$ simultaneously reverse their movement, the claim of the lemma holds by induction on the number of bounces. □

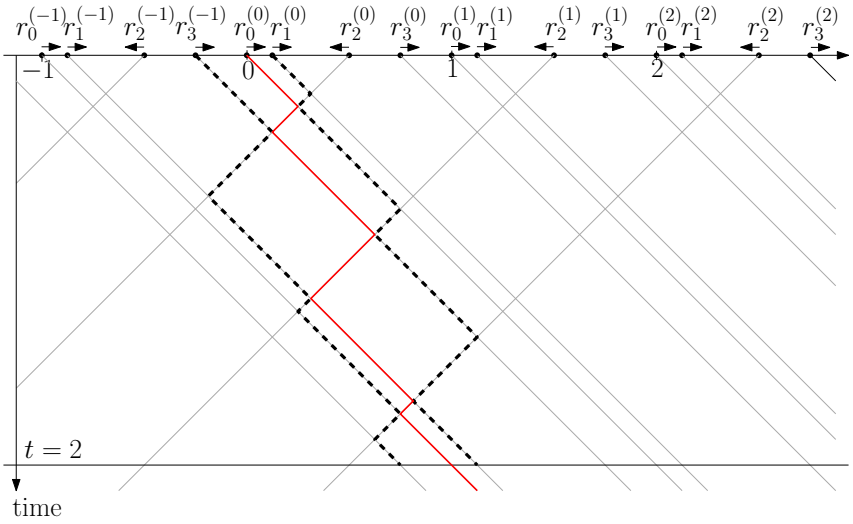

Figure 1. Example of a bouncing movement of four robots

We use the concept of a *baton*, applied recently in [30]. Suppose that each robot initially has a virtual object (baton), that the robot carries during its movement, but at the moment of meeting, two robots

exchange their batons. By $b_i^{(j)}$ we denote the baton originally held by robot $r_i^{(j)}$ and by $b_i^{(j)}(t)$ we denote the position of this baton on the infinite line at time t. We can easily show the following lemma.

Lemma 2. *For all $t \geq 0, 0 \leq i \leq n-1$ and $j \in \mathbb{Z}$ we have $b_i^{(j)}(t) = b_i^{(j)}(0) + dir_i \cdot t = b_i^{(0)}(0) + j + dir_i \cdot t$.*

Proof. Since the bouncing robots exchange their batons, the batons travel at constant speed 1 in their original directions. Therefore, at time t each baton travelled the distance t so we have $b_i^{(j)}(t) = b_i^{(j)}(0) + dir_i \cdot t$. On the other hand, by construction we have $b_i^{(j+1)}(0) = b_i^{(j)}(0) + 1$ and both batons $b_i^{(j)}, b_i^{(j+1)}$ travel at unit speed in the same direction. Hence, we have by induction on j, that $b_i^{(0)}(t) = b_i^{(j)}(t) + j$. The claim of the lemma follows. □

In Fig. 1 the trajectories of all the batons held originally by the robots going in direction *dir* are the lines of slope *dir*. Each robot r_i bounces while its trajectory intersects a trajectory of some baton, since this baton is then held by one of the robots r_{i-1}, r_{i+1}. For example, the trajectory of robot $r_0^{(0)}$, is represented by a fat polyline on Fig. 1, while the trajectories of its neighbor robots $r_3^{(-1)}$ and $r_1^{(0)}$ bouncing at $r_0^{(0)}$ are given by dashed polylines.

Lemma 3. *Consider robot r_a, which at the time moment t, while traveling in direction dir, meets some other robot. Suppose that, at the time of this meeting, r_a travelled the total distance d in direction dir (hence the total distance of $t-d$ in direction $-dir$). Then there exists a robot r_b, which was originally positioned at distance $(2d \mod 1)$ in direction dir on the ring. More precisely, $(2d \mod 1) = dist(r_a, r_b)$ if $dir = 1$ and $(2d \mod 1) = dist(r_b, r_a) = 1 - dist(r_a, r_b)$ if $dir = -1$. Moreover r_b started its movement in direction $-dir$.*

Proof. Suppose that at time t robot r_a traveling in direction *dir* meets some other robot traveling in the opposite direction (e.g. on Fig. 1, see the intersection of the trajectory of $r_0^{(0)}$ with the trajectory of the baton $b_2^{(2)}$ originally held by $r_2^{(2)}$). Suppose that the baton obtained by r_a at the moment of the meeting was originally held by some robot r_b. Robot r_a travelled the total distance d in direction *dir* and the total distance $t-d$ in direction $-dir$, while the baton obtained by r_a at the moment of the bounce travelled distance t in direction $-dir$. Hence during time $t-d$ robot r_a and the baton stayed at the same distance and during time d they were both traveling approaching each other (i.e. jointly covering total distance $2d$ while approaching). Therefore, at time $t = 0$ the distance between robots r_a and r_b was $2d$. Since r_b may be a copy of a robot and all copies of the same robot are at integer distance, the distance of r_a to r_b on the ring is $2d \mod 1$. The initial direction of r_b equals the direction of its original baton, i.e. $-dir$. □

Remark 1. The value $(2d \mod 1)$ may sometimes be equal to zero which corresponds to r_a meeting the robot currently holding the original baton of r_a (e.g. the sixth bounce of $r_0^{(0)}$ on Fig. 1). On the other hand, some meetings of robots may correspond to the same computed value of $(2d \mod 1)$ (e.g. all odd-numbered bounces of $r_0^{(0)}$ on Fig. 1), so some bounces do not have a new informative value about other robot positions.

The algorithm `RingBounce` executed by a robot, which reports initial positions and directions of all other robots on the ring, uses Lemma 3. Each bounce results in the output of information concerning one robot of the ring. In this way, a robot running such an algorithm needs only a constant-size memory. An additional test is made in line 10 to avoid outputting the same robot position more than once.

The robot's memory consists of two real variables *right* and *left* in which the robot will store the total distance travelled, respectively, in the counterclockwise and clockwise direction. The robot also accesses its system variable *clock* which automatically increases proportionally to the time spent while traveling (i.e. to the distance travelled).

Theorem 1. *Suppose that among all robots bouncing on the ring there is at least one robot having initial clockwise direction and at least one robot with the initial counterclockwise direction. The algorithm* `RingBounce`*, executed by any robot of the collection, correctly reports the initial positions and directions of all robots on the ring with respect to its initial position.*

Algorithm `RingBounce` (*dir* : $\{-1, 1\}$);

```
1.   var left ← 0, right ← 0 : real;
2.   reset clock to 0;
3.   while true do
4.      do walk in direction dir until
5.           ((clock − left ≥ 1/2) and (clock − right ≥ 1/2)) or a meeting occurs;

6.      if (clock − left ≥ 1/2) and (clock − right ≥ 1/2) then EXIT;
7.      if dir = 1 then
8.           right ← clock − left;
9.           if (0 < right < 1/2) then
10.               OUTPUT robot at original position 2 · right and direction −dir;
11.     else
12.          left ← clock − right;
13.          if (0 < left < 1/2) then
14.               OUTPUT robot at original position 1 − 2 · left and direction dir;
15.     dir ← −dir;
```

Proof. Suppose w.l.o.g., that the robot executing `RingBounce` is robot r_0. Since there exists at least one other robot starting in the direction different from dir_0, robot r_0 will alternately travel in both directions, indefinitely bouncing against its neighbors r_1 and r_{n-1} on the ring.

We show by induction, that at the start of each iteration of the while loop from line 3, the variable $left$ (resp. $right$) equals the total distance travelled by r_0 clockwise (resp. counterclockwise). Suppose, by symmetry, that r_0 walks counterclockwise in the i-th iteration and the inductive hypothesis is true at the start of this iteration. Since, by inductive hypothesis, variable $left$ keeps the correct value through i-th iteration, variable $right$ is correctly modified at line 8, as the $clock$ value equals the total distance travelled in both directions. Consequently, the inductive claim is true in the $(i+1)$-th iteration.

We prove now that positions and directions of all robots are correctly reported before the algorithm ends. Take any robot r_i, $1 \leq i \leq n-1$. We consider first the case when the initial direction of r_i was clockwise. The trajectory of its original baton $b_i^{(0)}$ is then a line of slope 1 (cf. Fig. 1). Observe that robot r_0 stays at the same distance from baton b_i when walking in the clockwise direction and approaches it (reducing their counterclockwise distance $dist(r_0, b_i)$) when walking counterclockwise. Since $dist(r_0, b_i) \leq 1$, and r_0 and b_i walk towards each other, they approach at speed 2 during the counterclockwise movement of r_0. Consequently, the trajectories of r_0 and b_i intersect and r_0 eventually meets robot r_1 carrying baton b_i. Indeed, in line 4 of algorithm `RingBounce`, robot r_0 continues its movement as long as its total distance travelled in the counterclockwise direction is less than 1/2, which leads to the meeting of r_0 and r_1 (carrying baton b_i), before both robots finish their executions of the algorithm. Consequently, at the moment of their meeting, r_0 outputs at line 10 the initial distance between $r_0^{(0)}$ and $r_i^{(0)}$ on line L, which equals twice the time spent while the robots were approaching each other. As r_0 may obtain a copy of the same baton more than once (cf. r_0 intersecting several trajectories of batons $b_2^{(j)}$ on Fig. 1), the condition $(0 < right < 1/2)$ at line 9 permits to report the position of each other robot once only. Indeed, only $r_i^{(0)}$ - the copy of r_i at the closest counterclockwise distance to r_0 verifies this condition.

Consider now the case when robot r_i, $1 \leq i \leq n-1$, starts its walk on the ring in the counterclockwise direction. Then r_0 obtains baton b_i while walking clockwise, i.e. at the moment of some bounce at r_{n-1}, while r_{n-1} holds baton b_i. In this case, robot r_0 stays at the same distance from baton b_i when walking in counterclockwise direction and approaches it (reducing their distance of $dist(b_i, r_0) = 1 - dist(r_0, b_i)$) when walking clockwise. At the moment when r_0 meets r_{n-1} holding baton b_i (whose trajectory originates from segment $[-1, 0]$ of L) the value of variable $left$ equals half the clockwise distance from $r_0(0)$ to $r_i(0)$.

Indeed, at the moment of the meeting, half of this distance was covered by r_0 walking clockwise (the value of $left$) and the other half was covered by the counterclockwise move of baton b_i. Consequently the clockwise distance from the initial position of r_0 to the initial position of r_i equals $1-2\cdot left$, correctly output at line 14. □

Observe that, once the original positions and directions of all robots are reported, it is easy to monitor all further movements of all robots of the collection, i.e. their relative positions at any moment of time. However, this would require a linear memory of the robot performing such task.

4. The Execution Time of Bouncing on the Ring

As stated in the introduction, we look for the algorithm of the optimal cost, i.e. the smallest possible total distance travelled, needed to correctly report any initial configuration. We show that the algorithm `RingBounce` is the optimal one, i.e. that the time moment, at which the robot can be sure that the positions of all other robots have been reported, is the time when the robot stops executing `RingBounce`. Observe that algorithm `RingBounce` has cost at least 1, i.e. a robot executing it must travel at least distance 1. Indeed, the loop from lines 4-5 continues unless robot's walk distance in each direction totals at least half the size of the ring. On the other hand, the example from Fig. 1 shows, that if the number of robots starting their walks in one direction is different from the number of robots starting walking in the opposite direction, the total cost of `RingBounce` may be higher. We have

Theorem 2. *Consider a collection of n robots on the ring, such that k of them, $1 \le k \le n/2$, have one initial direction and the remaining $n-k$ robots have the other initial direction. Then the cost of* `RingBounce` *is $C_{\mathcal{RB}}(n,k) \le \frac{1}{2}\lceil \frac{n}{k} \rceil$.*

Proof. If there are more robots starting in one direction, say positive direction dir, than in direction $-dir$ then r_i gets more frequently dir-moving batons (cf. Fig. 1). Since the route of r_i, intersects the trajectory of each baton only once, r_i must meet copies of batons originating from other segments than $[0,1]$ of line L. By counting we show that the last such segment is $[\lceil (n-k)/k \rceil - 1, \lceil (n-k)/k \rceil]$. Hence, in the worst case, r_i walks distance $1/2$ in direction dir and distance $\lceil (n-k)/2k \rceil$ in direction $-dir$.

By Lemma 1, we can translate the collection of robots and start their enumeration so that any of them is at the point 0 of line L and, w.l.o.g., it is sufficient to consider the total walk length of r_0. By symmetry we assume that r_0 starts walking counterclockwise on the ring.

Consider first the case when $n-k$ robots from the claim of the theorem start walking counterclockwise and k robots start walking clockwise on the ring, with $k \le n-k$. Note that r_0 alternately changes its direction of walk and, according to lines 4-5 of algorithm `RingBounce`, it has to travel a distance of at least $1/2$ in each direction. At the conclusion of each segment of the clockwise walk around the ring (i.e. left walk along line L), r_0 bounces against r_{n-1}, collecting one of the $n-k$ batons traveling counterclockwise. Denote by t_{2i}, for $i = 1, \ldots, n-k$ the sequence of the consecutive time moments of all bounces of r_0 against robot r_{n-1} (recall that time equals the total distance travelled up to that moment). Suppose that r_0 starts executing algorithm `RingBounce` at time $t_0 = 0$ and denote by t_{2i+1}, for $i = 0, \ldots, n-k-1$ the sequence of the consecutive time moments of all bounces of r_0 against robot r_1. At time $t_{2(n-k)}$, r_0 gets the originally held baton b_0 and the total length of its clockwise travel becomes exactly $1/2$ (i.e. the value of variable $left$ becomes $1/2$). Since $t_1 < t_2 < \ldots < t_{2(n-k)}$, before time $t_{2(n-k)}$ r_0 bounced also $n-k$ times against r_1, each time getting a baton, which is traveling clockwise. If $k = n/2$, there are $k = n-k$ lines of slope 1 originating from segment $[0,1)$ of L, which are trajectories of k batons traveling clockwise, see Fig. 2. Therefore, the loop from lines 4-5 of algorithm `RingBounce` continues until variable $right$ equals $1/2$ and algorithm finishes through the exit condition at line 6. In this case the total walk time equals to $left + right = 1$ and $\frac{1}{2}\lceil \frac{n}{k} \rceil = 1$ so the claim of the theorem holds.

In the case $k < n/2$ there are only k batons traveling clockwise ($k < n-k$), so some of them are received more than once by r_0 during the bounces at times $t_1, t_3, \ldots, t_{2(n-k)-1}$. Therefore, only k copies of batons traveling clockwise originate from each integer segment $[i, i+1)$ on line L. Consequently, r_0 obtains batons whose trajectories originate from segments other than $[0,1)$ and its total traveling distance in the

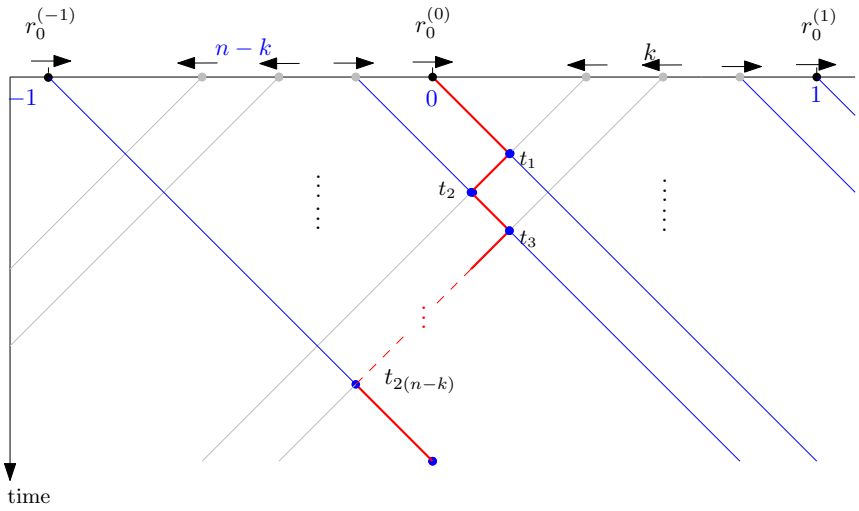

Figure 2. $n-k$ batons travel in counterclockwise direction while k batons travel in clockwise direction, the trajectory of r_0 appears in red. By time $t_{2(n-k)}$, robot r_0 has discovered all $n-k$ batons traveling in counterclockwise direction as well as those traveling in clockwise direction

counterclockwise direction exceeds 1/2. More precisely, the $(n-k)$-th consecutive copy of a baton traveling clockwise, obtained at time $t_{2(n-k)-1}$ must originate from the segment $[i^*-1, i^*]$, where $i^* = \lceil \frac{n-k}{k} \rceil$. The distance from the initial position of r_0 and the initial position of the baton met at time $t_{2(n-k)-1}$ does not exceed the value of $\lceil \frac{n-k}{k} \rceil$. Since robot r_0 has to travel counterclockwise at most half of this distance (the other half being covered by the moving baton), the total time spend by r_0 in both directions does not exceed

$$1/2 + \frac{1}{2}\left\lceil \frac{n-k}{k} \right\rceil = \frac{1}{2}\left\lceil \frac{n}{k} \right\rceil$$

Consider now the second case in which $n-k$ robots from the claim of the theorem start walking clockwise and k robots start walking counterclockwise on the ring, with $k < n-k$. As in the previous case we can denote by $t_1 < t_2 < \ldots < t_{2(n-k)-1}$ the consecutive bounce times, where t_i for odd values of i denote the times of the bounces of r_0 against r_1 and those for even values of i denote the times of the bounces against r_{n-1}. By symmetry to the first case, the bounce at time $t_{2(n-k)-1}$, when r_0 moves counterclockwise, arises when the variable *right* does not exceed the value of 1/2 (i.e. the total distance travelled counterclockwise by r_0) and the value of *left* is already greater than 1/2. At this time moment, robot r_0 goes clockwise and after the last bounce at time $t_{2(n-k)}$ continues counterclockwise and exits algorithm `RingBounce` where variable *right* becomes equal to 1/2. Indeed, since only $n-k$ batons travel clockwise, the next bounce of robot r_0 would imply getting a baton whose trajectory originates at segment $[1, 2]$ of line L, but this would make variable *right* exceed first the value of 1/2 and cause the exit in line 6 of `RingBounce`.

Similarly to the previous case, as $k < n-k$, at some of the bounces at times $t_2, t_4, \ldots, t_{2(n-k)}$, robot r_0 obtains the same batons. More precisely, during the bounce at time $t_{2(n-k)}$ r_0 obtains the baton whose trajectory originates at segment $[-\lceil \frac{n-k}{k} \rceil, -\lceil \frac{n-k}{k} \rceil + 1]$ of line L. Hence the total clockwise distance travelled by r_0 does not exceed $\frac{1}{2}\lceil \frac{n-k}{k} \rceil$ and the total distance travelled in both directions does not exceed $1/2 + \frac{1}{2}\lceil \frac{n-k}{k} \rceil = \frac{1}{2}\lceil \frac{n}{k} \rceil$ proving the claim of the theorem. □

From Theorem 2 we immediately have the following corollary, which bounds the worst-case walking time for a robot.

Corollary 1. *Assuming that the collection of n robots admits robots starting their movements in both directions around the ring, Then the cost of* `RingBounce` *is* $C_{\mathcal{RB}}(n) \leq \frac{n-1}{2}$.

The algorithm `RingBounce` continues until the total lengths of walks in both directions reach the values of at least 1/2, since this guarantees that the presence of each robot is eventually detected. The following

theorem proves that the cost of `RingBounce` algorithm is optimal even if the (a priori) knowledge of the number of robots is assumed.

Theorem 3. *Suppose that there is a collection of n robots on the ring, such that k of them, $1 \leq k < n/2$, have one initial direction and the remaining $n-k$ robots have the other initial direction. Then for every $\epsilon > 0$ there exists a distribution of such robots on the ring with their initial positions $0 \leq r_0 < r_1 < \ldots < r_{n-1} < 1$, so that a position detection algorithm terminating at time $\frac{1}{2}\lceil \frac{n}{k} \rceil - \epsilon$ cannot determine the initial positions of all robots on the ring, even if the values of n and k are known in advance.*

Proof. Consider the following collection of n robots $r_0, r_1, \ldots, r_{k-1}, r_k, \ldots, r_{n-1}$ on the ring, where each r_i has a counterclockwise starting direction for $0 \leq i < k$ and the clockwise starting direction for $k \leq i \leq n-1$, with the initial positions of the robots

$$r_0 = 0, r_1 = \frac{\epsilon}{2^{k-1}}, r_2 = \frac{\epsilon}{2^{k-2}}, \ldots, r_{k-1} = \frac{\epsilon}{2},$$
$$r_k = 1 - \frac{\epsilon}{2}, \ldots, r_{n-1} = 1 - \frac{\epsilon}{2^{n-k}}$$

Suppose, by contradiction, that a position detection algorithm executed by robot r_0 can determine the positions of all other robots with the total walking distance of r_0 at most $\frac{1}{2}\lceil \frac{n}{k} \rceil - \epsilon$. Robot r_0, in order to determine positions of all other robots, has to obtain each baton $b_1, \ldots, b_{n-1}$. Robot r_0 gets the clockwise-traveling batons $b_k, b_{k+1}, \ldots, b_{n-1}$ in this order at the moments of its bounces against r_1. On the other hand, the remaining batons are obtained by r_0 in order $b_{k-1}, b_{k-2}, \ldots, b_1$ at the moments of its bounces against r_{n-1}. However, since $k < n$, at least some of the batons $b_{k-1}, b_{k-2}, \ldots, b_1$ are obtained repeatedly (in the same cyclic order) because the left and right bounces are alternated. More precisely, baton b_{k-1} is obtained $\lceil (n-k)/k \rceil$ times by b_0, hence this sequence of batons is

$$\overleftarrow{b_k}, \overrightarrow{b_{k-1}}, \overleftarrow{b_{k+1}}, \overrightarrow{b_{k-2}}, \ldots, \overleftarrow{b_{2k-1}}, \overrightarrow{b_0}, \overleftarrow{b_{2k}}, \overrightarrow{b_{k-1}},$$
$$\overleftarrow{b_{2k+1}}, \overrightarrow{b_{k-2}}, \ldots, \overleftarrow{b_{3k-1}}, \overrightarrow{b_0}, \ldots, \overleftarrow{b_{n-1}}, \overrightarrow{b_f}$$

where $f = k(\lceil n/k \rceil + 1) - (n+1)$ and $\overleftarrow{b_i}$ denotes baton b_i traveling clockwise and $\overrightarrow{b_j}$ denotes baton b_j traveling counterclockwise. The copies of the last two batons of this sequence are the most distant from r_0 on line L. The trajectory of baton $\overleftarrow{b_{n-1}}$ originates in segment $[0,1)$ of line L and $dist(r_0^0(0), b_{n-1}^0(0)) = 1 - \frac{\epsilon}{2^{n-k}}$. On the other hand, the trajectory of baton $\overrightarrow{b_{k(\lceil n/k \rceil + 1) - (n+1)}}$ obtained by b_0 starts in the segment $[-\lceil (n-k)/k \rceil, -\lceil (n-k)/k \rceil + 1]$ and its original distance to b_0 is

$$dist(b_f^{(-\lceil (n-k)/k \rceil)}(0), b_0^{(0)}(0)) > \lceil (n-k)/k \rceil - \frac{\epsilon}{2}$$

As in order to meet each of these batons, r_0 has to travel half of its original distance to each of them (the other half is covered by the corresponding baton itself) the total travel time by r_0 is bound by

$$\frac{1}{2}(1 - \frac{\epsilon}{2^{n-k}}) + \frac{1}{2}(\lceil (n-k)/k \rceil - \frac{\epsilon}{2}) >$$
$$\frac{1}{2}(1 + \lceil (n-k)/k \rceil - \epsilon) > \frac{1}{2}(\lceil n/k \rceil) - \epsilon$$

which contradicts the assumed claim and proves the theorem. □

Clearly each configuration of robots with the same initial direction of all robots is infeasible, because no robot ever bounces. Consequently from Theorem 2 and Theorem 3 follows

Corollary 2. *The family of infeasible initial configurations of robots on the ring contains all configurations with the same initial direction of all robots.* `RingBounce` *is the optimal position detection algorithm for all feasible initial configurations of robots on the ring. This algorithm assumes constant-size memory of the robot running it.*

Clearly, we can easily adapt algorithm `RingBounce`, so for infeasible initial configuration the algorithm stops and reports the infeasibility. It is sufficient to test whether the very first walk of the robot ends with a bounce before the robot traverses the distance of 1/2.

5. Bouncing on the Line Segment

In this section we show how the algorithm for bouncing robots may be used for the case of a segment. We suppose that each robot walks along the unit segment changing its direction when bouncing from another robot or from an endpoint of the segment. Robots have the same capabilities as in the case of the ring and they cannot distinguish between bouncing from another robot and bouncing from a segment endpoint.

We consider the segment $[0,1)$ containing n robots, initially deployed at positions

$$0 \leq r_0(0) < r_1(0), \ldots, r_{n-1}(0) < 1.$$

Each robot r_i, $0 \leq i \leq n-1$ is given an initial direction dir_i, such that $dir_i = 1$ denotes the left to right initial movement and $dir_i = -1$ denotes initial movement from right to left on segment $[0,1)$. The robots start moving with unit speed at the same time moment $t = 0$ at the predefined directions and they change direction upon meeting another robot or bumping at the segment endpoint. The main difficulty of the segment case is that the robot r executing the position detection algorithm for the ring has to report the *relative* locations of other robots, i.e. their distances to its own initial position $r(0)$, while in the segment case the *absolute* distance from $r(0)$ to the segment endpoint has to be found.

We show in this section that the bouncing problem is feasible for all initial robot configurations except a small set of symmetric ones. Intuitively, an initial configuration of robots is *symmetric* if the unit segment may be partitioned into k subsegments $S_0, S_1, \ldots, S_{k-1}$, such that the positions and directions of robots in each subsegment form a reflected copy of positions and directions of robots in a neighboring subsegment (see Fig. 3). More formally we have the following

Definition 1. *A configuration* $C = ((r_0(0), dir_0), \ldots, (r_{n-1}(0), dir_{n-1}))$ *is symmetric if there exists a positive integer* $k < n$, *such that* $n \mod k = 0$ *and the partition of segment* $S = [0,1)$ *into subsegments* $S_0 = [0, \frac{1}{k})$, $S_1 = [\frac{1}{k}, \frac{2}{k}), \ldots, S_1 = [\frac{k-1}{k}, 1)$ *with the following property. For each robot* r_i, $0 \leq i < n$, *if* $r_i(0) = \frac{p}{n} + x$, *for* $0 \leq x < \frac{1}{k}$, *(i.e.* $r_i(0) \in S_p$*),* $0 \leq p < n$, *then, if* $p > 0$, *there exists a robot* $r_{i'}$, *such that* $r_{i'}(0) = \frac{p}{n} - x$ *and* $dir_{i'} = 1 - dir_i$ *and, if* $p < n-1$, *there exists a robot* $r_{i''}$, *such that* $r_{i''}(0) = \frac{p+2}{n} - x$ *and* $dir_{i''} = 1 - dir_i$.

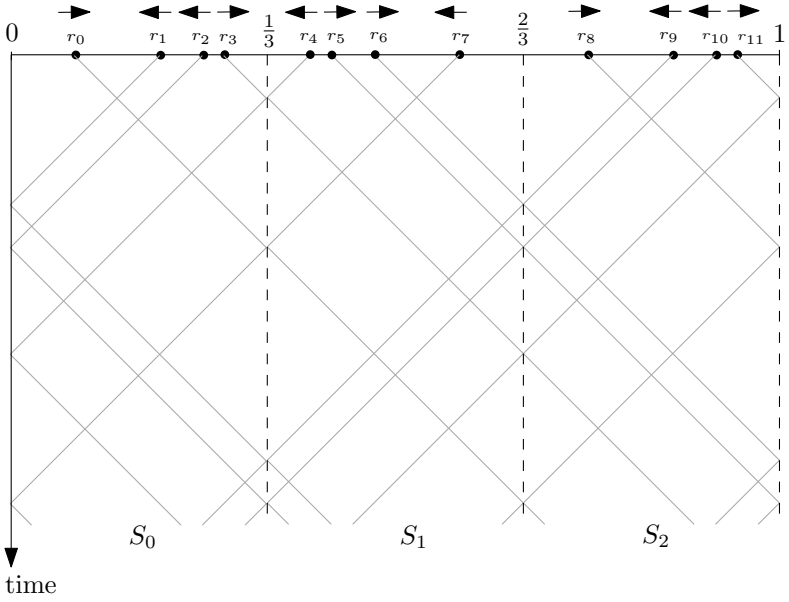

Figure 3. Example of a symmetric initial configuration of $n = 12$ robots containing $k = 3$ subsegments

Theorem 4. *Every symmetric initial configuration of robots is infeasible.*

Proof. Let $C_1 = ((r_0(0), dir_0), \ldots, (r_{n-1}(0), dir_{n-1}))$ be a symmetric configuration and k the number of consecutive subsegments, each one being the reflected copy of its neighbor. Construct now configuration $C_2 = ((r'_0(0), dir'_0), \ldots, (r'_{n-1}(0), dir'_{n-1}))$ of n robots considering also a sequence of n equal size intervals and swapping the roles of odd-numbered and even-numbered robots of C_1. More precisely for each robot r_i, such that $r_i(0) \in S_p = [\frac{p}{n}, \frac{p+1}{n})$ there exists a robot r'_j such that $r'_j(0) = \frac{2p+1}{n} - r_i(0)$ and $dir'_j = 1 - dir_i$. Observe that, no robot ever crosses the boundary of any subsegment S_p, i.e. $r_i(0) \in S_p$ implies $r_i(t) \in S_p$, for any $t \geq 0$. Indeed, by construction, for any robot reaching and endpoint of S_p, different from points 0 and 1, at the same time moment there is another robot approaching this endpoint from the other side within the reflected copy of S_p provoking a bounce (cf. Fig. 4). Therefore, within each even-numbered subsegment S_{2n} of a symmetric configuration the relative positions of robots and their directions are the same (similarly within each odd-numbered subsegment). Consequently, no robot can distinguish whether it is, say, in an even-numbered segment of C_1 or in an odd-numbered segment of C_2 so its position in segment $[0, 1)$ is unknown. □

We show now how the position detection algorithm for the ring may be used in the case of the segment.

Let S be a unit segment containing n robots at initial positions $r_0(0) < r_1(0) < \ldots < r_{n-1}(0)$ and the initial directions $dir_0, \ldots, dir_{n-1}$. Suppose that a segment $S^R \subset [1, 2]$ is the reflected copy of S containing n robots $r^R_n, \ldots, r^R_{2n-1}$ at the initial positions

$$r^R_n(0) = 2 - r_n(0) < r^R_{n-1}(0) = 2 - r_{n-1}(0) < \ldots < r^R_0(0) = 2 - r_0(0).$$

The initial directions of each robot r^R_i is $1 - dir_i$ for $0 \leq i < n$. Let R_2 be the ring of perimeter 2 composed of segment S concatenated with segment S^R, with points 0 and 2 identified.

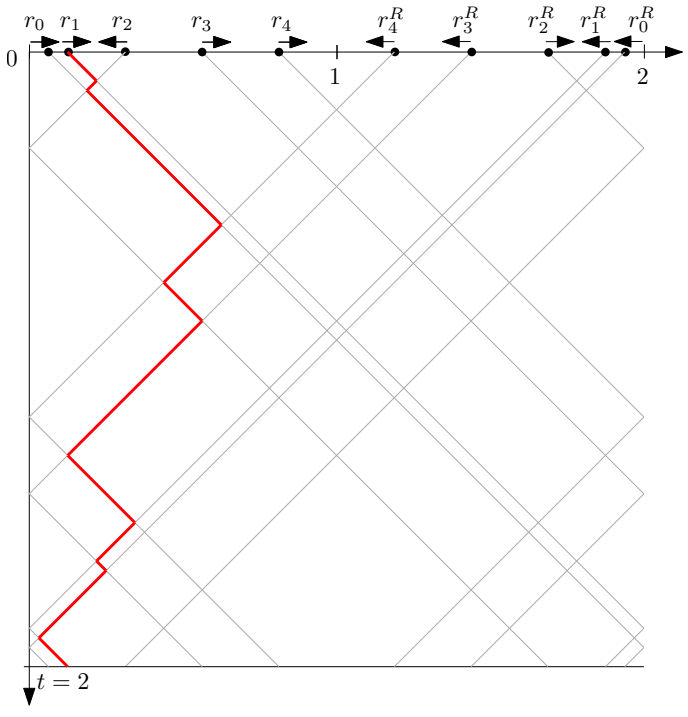

Figure 4. Five robots on a segment $[0, 1)$ and their reflected copy

Consider the walk of robots r_i, for $0 \leq i < n$, within segment S and ring R_2. Let $t_0 = 0$ and $0 \leq t_1 < t_2 \ldots$ be the sequence of time moments during which some bounces occur. Each such bounce takes place either between some pair of robots or when some robot bounces from an endpoint of S. It is easy to see by induction on i that at any time moment $t \in [t_i, t_{i+1}]$ each robot r_j has the same positions in S and R_2 as

well as the same direction of movement and that the S^R part of R_2 is a reflected copy of S. Indeed, by construction, this condition is true for the interval $[t_0, t_1]$. If robots r_j, r_{j+1} bounce against each other in S at time t_i, at the same time robots r_j, r_{j+1} bounce in R_2, as well as, by symmetry r_j^R bounces against r_{j+1}^R. If in time t_i robot r_0 (or r_{n-1}) bounce from an endpoint of S, by inductive hypothesis r_0 bounces against r_0^R at point $0 \in R_2$ (or r_{n-1} bounces against r_{n-1}^R at point $1 \in R_2$). In each case, the inductive condition holds. We just showed

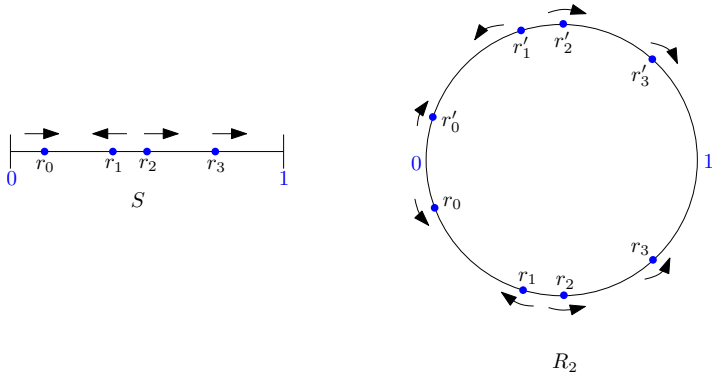

Figure 5. Example of the corresponding ring, R_2, of a segment, S, of lenght 1 wherein `RingBounce` may be used to find the initial positions of the robots on S

Lemma 4. *The bounce sequence of any robot r_i on segment S is the same as the bounce sequence of r_i on ring R_2.*

To prove that only symmetric configurations of robots on the segment are infeasible we need the following lemma.

Lemma 5. *Suppose that the initial configuration of robots $C = ((r_0(0), dir_0), \ldots, (r_{n-1}(0), dir_{n-1}))$ on a unit segment is not symmetric. Then no internal robot r_i, for $1 \leq i \leq n-2$, may have all its left bounces or all right bounces at the same point of the unit segment.*

Proof. Suppose, by contradiction that there exists an internal robot having all its left bounces at the same point (the proof in the case of all right bounces falling at the same point is similar, by symmetry). Let i be the smallest index $1 \leq i \leq n-2$ of a robot with this property and point x, $0 < x < 1$, be the point of all left bounces of r_i. We show first that the initial configuration of robots belonging to segment $[0, x]$ is the reflected copy of the initial configuration of robots belonging to segment $[x, 2x]$ Then robot r_{i-1} has all its right bounces also at point x. Consequently, at each moment of time after the first such bounce, the position and the direction of robot r_{i-1} is a symmetric (reflected) copy of robot r_i with respect to point x. Then, if $i \geq 2$, the trajectory of r_{i-2} is a reflected copy of the trajectory of r_{i+1}. By induction on i, for any $q \geq 0$ the trajectory of r_q is the reflected copy of the trajectory of r_{2i-q-1} and finally the trajectory of r_0 is the reflected copy of the trajectory of r_{2i-1}. Therefore, all right bounces of robot r_{2i-1} are at point $2x$ of the unit segment, so initial configuration of robots belonging to segment $[0, x]$ is the reflected copy of the initial configuration of robots belonging to segment $[x, 2x]$, as needed.

By induction on j we prove that each subsegment $[(j-1)x, jx]$ is a reflected copy of subsegment $[jx, (j+1)x]$. By minimality of x, no such subsegment contains a point which is never crossed by any robot, hence, for some value of j, we have $jx = 1$, concluding the proof. □

We can show now, that the set of configurations on the unit segment for which no position detection algorithm exists is exactly the set of symmetric configurations. For all other configurations we propose an optimal position detection algorithm. We suppose that the robot assumes that its initial direction on the segment is positive. Otherwise, the robot needs to be *chirality aware*, i.e. capable of identifying the positive direction of the segment.

Theorem 5. *For any collection of n robots not in a symmetric initial configuration on the unit segment there exist a position detection algorithm $\mathcal{A}$ with cost $C_{\mathcal{A}}(n) = 2$. For any $\epsilon > 0$ there exist collections of robots, such that some of them cannot terminate the execution of any position detection algorithm before time $2 - \epsilon$.*

Proof.

To construct algorithm $\mathcal{A}$ adapt algorithm `RingBounce` in the following way. The constant of $1/2$ used in lines 5, 6, 9 and 13 is changed to 1. Moreover, the values of original positions output in lines 10 and 14 are multiplied by a factor of 2 and put in the list C instead of being directly output. By Lemma 4 the algorithm finds the positions of $2n$ robots of ring R_2 constructed from segment S. Note that, as ring R_2 has size 2, we needed to scale up the time and distance constants of the algorithm by the factor of 2.

Since the robots of S are not in a symmetric initial configuration, by Lemma 5, only the endpoints of S are the points which are never crossed by any robot in S. Consequently there are only two points in R_2, which are never crossed by any robot. This unique pair of (antipodal) points split ring R_2 into two segments, segment S, and it reflected copy S^R. Since the positions of all robots are stored in list C, it is possible to perform in algorithm $\mathcal{A}$ the generation of the bounce sequence of each robot of C, in order to find which two robots bounce in one direction against the same positions of the ring. This way, the first and the last robot on the unit segment as well as its left and right endpoints may be identified, which permits to determine the rank and the absolute initial position of the robot running the algorithm, as well as those of all other robots.

In order to analyze the cost of such algorithm, observe that, since exactly half of $2n$ robots in R_2 have the same initial direction, by Theorem 2, robot r_i terminates its walk at time $\frac{1}{2}\lceil\frac{2n}{n}\rceil \cdot c = 2$, where $c = 2$ is the scaling factor.

We prove now the second part of the claim of the theorem. For any $\epsilon > 0$ we construct two different configurations of robots C_1, C_2 on the unit segment, such that for some robots from C_1 and C_2 the bounce sequence until time $2 - \epsilon$ is the same. Consequently, the robot observing such bounce sequence cannot unambiguously report positions of other robots.

Let C_1 be the configuration of two robots r_0, r_1, such that $dir_0 = dir_1 = 1$ and $r_0(0) = \frac{\epsilon}{5}, r_1(0) = \frac{2\epsilon}{5}$. We find below the first two values t_1, t_2 of the bounce sequence of robot r_0. Robot r_1 reaches point 1 of the segment and bounces at time $t^* = 1 - \frac{2\epsilon}{5}$, while robot r_0 is at point $1 - \frac{\epsilon}{5}$ of the segment. Since at time t^* the robots start to approach, they meet after additional time $\frac{\epsilon}{10}$, so $t_1 = 1 - \frac{2\epsilon}{5} + \frac{\epsilon}{10} = 1 - \frac{3\epsilon}{10}$ and $r_0(t_1) = 1 - \frac{\epsilon}{5} + \frac{\epsilon}{10} = 1 - \frac{\epsilon}{10}$. At time t_1 robot r_0 starts moving left on the segment until it bounces at its endpoint 0. This takes time $1 - \frac{\epsilon}{10}$, so $t_2 = t_1 + 1 - \frac{\epsilon}{10} = 2 - \frac{2\epsilon}{5} > 2 - \epsilon$.

Consider now configuration C_2, containing two robots r_0, r_1, such that $dir_0 = dir_1 = 1$ and $r_0(0) = \frac{\epsilon}{10}, r_1(0) = \frac{\epsilon}{2}$. The similar analysis reveals that $t^* = 1 - \frac{\epsilon}{2}$, $r_1(t^*) = 1$, and $r_0(t^*) = 1 - \frac{2\epsilon}{5}$. At time t^*, r_0 and r_1 start approaching and meet at time $t_1 = t^* + \frac{\epsilon}{5} = 1 - \frac{3\epsilon}{10}$, while $r_0(t_1) = 1 - \frac{\epsilon}{5}$. After the bounce at time t_1, r_0 walks left until it bounces at endpoint 0 at time $t_2 = t_1 + 1 - \frac{\epsilon}{5} = 2 - \frac{\epsilon}{2} > 2 - \epsilon$.

Since for both configurations C_1, C_2 we have $t_1 = 1 - \frac{3\epsilon}{10}$ and $t_2 > 2 - \epsilon$, hence robot r_0 cannot unambiguously output the initial robots' positions before time $2 - \epsilon$. □

As the algorithm for the segment, presented in the proof of Theorem 5 assumes storing in robot's memory the positions of all robots, from Theorems 4 and 5 follows

Corollary 3. *The family of infeasible initial configurations of robots on the segment contains all symmetric initial configurations of robots. There exists an optimal position detection algorithm for all feasible initial configurations of robots on the segment. This algorithm assumes $O(n)$-size memory of the robot executing it.*

6. Conclusion

The algorithms of the paper may be extended to the case when only one robot r_0 starts moving initially (while all other robot movements are triggered by bounces) and r_0 must report other robots' initial positions. Indeed, observe that all robots must be moving at no later than time 1 for the ring and at no later than time

2 for the segment. Robot r_0 may then compute the trajectories of all other robots as if they started moving simultaneously and then successively compute the sequence of motion triggering bounces of all robots. Other types of asynchrony would also be interesting to consider and analyze.

There are several interesting open problems resulting from our study. One is to determine whether there exists an optimal position detection algorithm for the segment using a constant size memory. Another is whether the bouncing problem may be solved for the case of robots having different initial speeds. If we assume the momentum conservation principle, so that the bouncing robots exchange their speeds, the baton trajectories still remain semi-lines of constant slopes. Therefore, if each robot is always aware of its current speed, perhaps it might be possible, that, after a finite time, it learns the starting positions and initial speeds of all other robots.

Another class of interesting open problems concerns extenions of our model to more general domains, robots as well as dynamics. It would be interesting to know whether our analysis can be extended to more general geometric graphs or even in 2D. In the latter case, one may ask whether it is possible to analyze the dynamics of the bouncing robots when they have physical constraints such as dimensionality (e.g., they are represented as discs), they can travel with not-necessarily identical speeds, and their bouncing dynamics obey laws of elasticity in classical and non-classical mechanics.

We also note that the location discovery performed by the collection of robots, presented in this paper, may be used for the equally-spaced self-deployment of the robots around the environment (e.g. to perform optimal patrolling) or for some other pattern formation task. However, such a task would require an additional robot capacity besides passive mobility the way it is assumed in this paper. Once the positions of the entire collection is known, the robots need to synchronize their movements, e.g. by adding waiting periods. In addition, the problem of information transmission and message passing between robots to maintain network connectivity and implement communication primimitives such as broadcasting and gossiping has been studied in the forthcoming publication [33].

Acknowledgements 1. Research of J. Czyzowicz and E. Kranakis supported in part by NSERC grants, L. Gąsieniec was sponsored by the Royal Society Grant IJP - 2010/R2 , O. Morales by MITACS grant and E. Pacheco by CONACyT and NSERC grant.

References

[1] I. Suzuki, M. Yamashita, Distributed anonymous mobile robots: Formation of geometric patterns, SIAM Journal on Computing 28 (4) (1999) 1347–1363.

[2] S. Das, P. Flocchini, N. Santoro, M. Yamashita, On the computational power of oblivious robots: forming a series of geometric patterns, in: Proceedings of the 29th ACM symposium on Principles of distributed computing, ACM, 2010, pp. 267–276.

[3] P. Flocchini, G.Prencipe, N. Santoro, P. Widmayer, Arbitrary pattern formation by asynchronous, anonymous, oblivious robots, Theoretical Computer Science 407 (1) (2008) 412–447.

[4] P. Flocchini, G. Prencipe, N. Santoro, Self-deployment of mobile sensors on a ring, Theoretical Computer Science 402 (1) (2008) 67–80.

[5] Y. Elor, A. M. Bruckstein, Uniform multi-agent deployment on a ring, Theoretical Computer Science 412 (8) (2011) 783–795.

[6] P. Flocchini, Uniform covering of rings and lines by memoryless mobile sensors, in: Encyclopedia of Algorithms, Springer, 2015.

[7] R. Kurazume, S. Nagata, S. Hirose, Cooperative positioning with multiple robots, in: Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on, IEEE, 1994, pp. 1250–1257.

[8] D. Fox, W. Burgard, H. Kruppa, S. Thrun, A probabilistic approach to collaborative multi-robot localization, Autonomous robots 8 (3) (2000) 325–344.

[9] R. Kurazume, S. Hirose, An experimental study of a cooperative positioning system, Autonomous Robots 8 (1) (2000) 43–52.

[10] S. I. Roumeliotis, G. A. Bekey, Distributed multirobot localization, Robotics and Automation, IEEE Transactions on 18 (5) (2002) 781–795.

[11] A. I. Mourikis, S. I. Roumeliotis, Performance analysis of multirobot cooperative localization, Robotics, IEEE Transactions on 22 (4) (2006) 666–681.

[12] A. Bahr, M. R. Walter, J. J. Leonard, Consistent cooperative localization, in: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE, 2009, pp. 3415–3422.

[13] K. Y. K. Leung, T. D. Barfoot, H. H. T. Liu, Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach, Robotics, IEEE Transactions on 26 (1) (2010) 62–77.

[14] S. Thrun, Robotic mapping: A survey, Exploring artificial intelligence in the new millennium (2002) 1–35.
[15] S. Kraus, O. Shehory, G. Taase, Coalition formation with uncertain heterogeneous information, in: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM, 2003, pp. 1–8.
[16] Y. Altshuler, A. M. Bruckstein, Static and expanding grid coverage with ant robots: Complexity results, Theoretical Computer Science 412 (35) (2011) 4661–4674.
[17] Y. Altshuler, V. Yanovski, I. A. Wagner, A. M. Bruckstein, Multi-agent cooperative cleaning of expanding domains, The International Journal of Robotics Research 30 (8) (2011) 1037–1071.
[18] I. Harmatia, K. Skrzypczyk, Robot team coordination for target tracking using fuzzy logic controller in game theoretic framework, Robotics and Autonomous Systems 57 (1) (2009) 75–86.
[19] G. Hollinger, S. Singh, J. Djugash, A. Kehagias, Efficient multi-robot search for a moving target, The International Journal of Robotics Research 28 (2) (2009) 201–219.
[20] J. Svennebring, S. Koenig, Building terrain-covering ant robots: A feasibility study, Autonomous Robots 16 (3) (2004) 313–332.
[21] S. Koenig, B. Szymanski, Y. Liu, Efficient and inefficient ant coverage methods, Annals of Mathematics and Artificial Intelligence 31 (1-4) (2001) 41–76.
[22] K. Sugihara, I. Suzuki, Distributed algorithms for formation of geometric patterns with many mobile robots, Journal of Robotic Systems 13 (3) (1996) 127–139.
[23] M. Yamashita, I. Suzuki, Characterizing geometric patterns formable by oblivious anonymous mobile robots, Theoretical Computer Science 411 (26) (2010) 2433–2453.
[24] R. Cohen, D. Peleg, Convergence properties of the gravitational algorithm in asynchronous robot systems, SIAM Journal on Computing 34 (6) (2005) 1516–1528.
[25] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Gathering of asynchronous robots with limited visibility, Theoretical Computer Science 337 (1) (2005) 147–168.
[26] R. Cohen, D. Peleg, Local spreading algorithms for autonomous robot systems, Theoretical Computer Science 399 (1) (2008) 71–82.
[27] H. Ando, Y. Oasa, I. Suzuki, M. Yamashita., Distributed memoryless point convergence algorithm for mobile robots with limited visibility, IEEE Transactions on Robotics and Automation 15 (5) (1999) 818–828.
[28] D. Angluin, J. Aspnes, Z. Diamadi, M. Fischer, R. Peralta, Computation in networks of passively mobile finite-state sensors, Distributed computing 18 (4) (2006) 235–253.
[29] D. Angluin, J. Aspnes, D. Eisenstat, Stably computable predicates are semilinear, in: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing, ACM, 2006, pp. 292–299.
[30] T. Friedetzky, L. Gasieniec, T. Gorry, R. Martin, Observe and remain silent (communication-less agent location discovery), in: Mathematical Foundations of Computer Science 2012, Springer, 2012, pp. 407–418.
[31] J. Czyzowicz, L. Gąsieniec, A. Kosowski, E. Kranakis, Boundary patrolling by mobile agents with distinct maximal speeds, in: Algorithms–ESA 2011, Springer, 2011, pp. 701–712.
[32] E. W. Dijkstra, Selected writings on computing: a Personal Perspective, Springer-Verlag New York, Inc., 1982.
[33] J. Czyzowicz, E. Kranakis, E. Pacheco, D. Pajak, Information spreading by mobile particles on a line (submitted for publication) (2015).